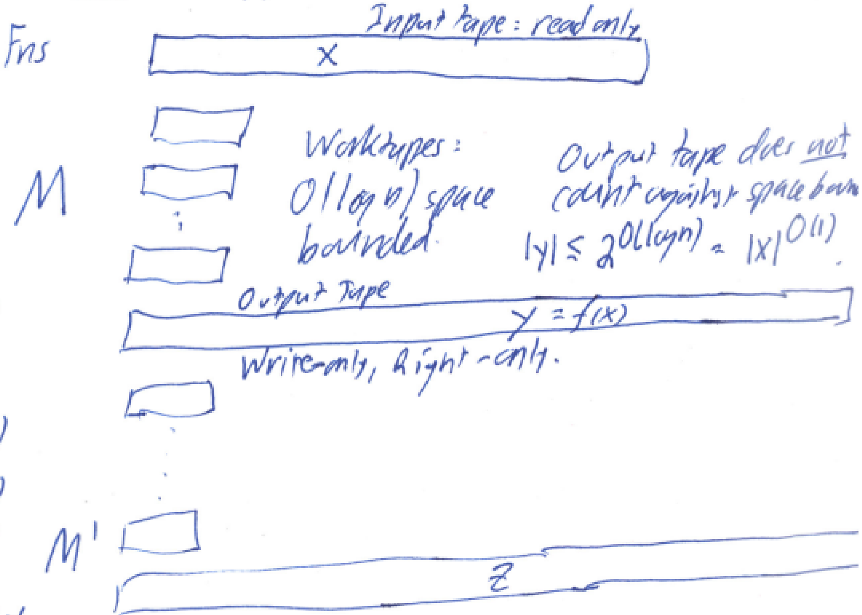
CSE491/596 Lecture Monday, Nov. 6: Completeness Under Logspace Reductions

<u>Log Space Reducibility</u> $\leq_m^{log}$ Finer than $\leq_m^p$ since $L \subseteq P$.

Log-Space Computable Fns



<u>Technical Point</u>: If f and g belong to <u>FL</u> (function logspace) then so does $g \circ f$. <u>Issue</u>: If M computes $y = f(x)$ and M' computes $z = g(y)$ we can "chain" M and M' together, but we <u>can't</u> <u>store</u> y, within the $O(\log|x|)$ space bd.

- If the input tapes of both machine are **right-only** as well as read-only, then there is no problem: the output $y = f(x)$ of M is streamed to M' computing $g(y) = z$ and never has to be written down.
- If each machine is allowed $r(n)$ left-to-right streaming passes over its input and y is a stream, then the tandem can operate with $r(n)^2$ passes on x.
- But if M' can demand to back up to a previous input bit y_{i-1} at any time, then we need to allow M to be restarted arbitrarily many times. This can be implemented by storing the current demand-bit i on another log-sized tape.

Whenever M' wants to move its input head <u>Left</u>, M <u>re-starts</u> from the beginning until it outputs bit <u>$i-1$</u> of y, which is stored. → $\square$ y_i $\boxed{i}$
If M' moves to $i+1$, M takes however long to output bit $i+1$. All the re-starting is inefficient for <u>time</u> but stays within $O(\log n)$ <u>space</u>.

Therefore $\leq_m^{log}$ reductions are transitive: $A \leq_m^{log} B \wedge B \leq_m^{log} C \Rightarrow A \leq_m^{log} C$.
In fact, every $\leq_m$ and $\leq_m^p$ reduction shown in the course has actually been a $\leq_m^{log}$ reduction or even the sharper one-pass streaming kind.

__Hallmarks of a $\leq_m^{\log}$ Reduction:__

- The objects it constructs have an explicit formula. E.g.:
  $G_\phi = (V_\phi, E_\phi)$, $V_\phi = \{x_i, \bar{x}_i : 1 \leq i \leq n\} \cup \{x_{ij}, \bar{x}_{ij} :$ variable x_i is in clause C_j, possibly negated$\}$
  $E_\phi = \{ \cdots \} \cup \{ \cdots \}$ etc.

- The individual items used in building G_ϕ etc. are finite clumps of $O(\log n)$-sized labels such as variable numbers i, clause #s j.

- (In consequence), __local__ features of the target object G_ϕ (or etc.) depend only on __local__ features of the source object (e.g., $C_1 \wedge \cdots \wedge C_m$) or on __simple global__ connections—like copying $\langle M, w \rangle$ or hooking up the B and G nodes in the __$3SAT \leq_m^{\log}$__ G3C example.

- All the NP-completeness results we've shown have been valid under $\leq_m^{\log}$.
- **GAP** is complete for NL under $\leq_m^{\log}$.
- The language **CVP** of the **Circuit Value Problem**: given a Boolean circuit C_n and an input $x \in \{0,1\}^n$, is $C_n(x) = 1$? is complete for P under $\leq_m^{\log}$
- The language **TQBF** of true **quantified Boolean formulas** is complete for PSPACE under $\leq_m^{\log}$. We will show this next.

Quantified Boolean Formulas (which are really logical sentences)

A **quantified Boolean formula** (QBF) may have quantifiers $\exists$ and $\forall$ on single Boolean variables as well as the Boolean connectives $\wedge$, $\vee$, $\neg$. A QBF ψ is in **prenex form** if it has the form

$$\psi = (Q_1 x_1)(Q_2 x_2) \cdots (Q_n x_n)\phi(x_1, x_2, \ldots, x_n),$$

where each Q_i is $\exists$ or $\forall$ and ϕ is an ordinary Boolean formula. The simplest example of a QBF in prenex form is

$$\psi = (\exists x_1)(\exists x_2) \cdots (\exists x_n)\phi(x_1, x_2, \ldots, x_n).$$

Then ψ is *true* if and only if ϕ is *satisfiable*. In musical counterpoint, the QBF

$$\psi = (\forall x_1)(\forall x_2) \cdots (\forall x_n)\phi'(x_1, x_2, \dots, x_n)$$

is true if and only if ϕ' is a tautology. Where it gets trickier---for our brains as well---is when the quantifiers **alternate** $\exists$ and $\forall$. Then the problem of whether a QBF is true evidently rises above the level of NP and co-NP. For a higher example from a game like chess, Black has a checkmate in three if

$$(\exists\, \overrightarrow{bm}_1)(\forall \overrightarrow{wm}_1)(\exists\, \overrightarrow{bm}_2)(\forall \overrightarrow{wm}_2)(\exists\, \overrightarrow{bm}_3) WhiteIsMated(\vec{\pi}''''').$$

Here the quantifiers read as being applied to possible moves in a chess position, but they are really running over Boolean variables

$$b_{1,1}, b_{1,2}, \dots b_{1,\ell}; w_{1,1}, w_{1,2}, \dots w_{1,\ell}; b_{2,1}, b_{2,2}, \dots b_{2,\ell}; w_{2,1}, w_{2,2}, \dots w_{2,\ell}; b_{3,1}, b_{3,2}, \dots b_{3,\ell}; \dots$$

that together code the possible moves in binary notation. In the background is another vector of variables $\vec{\pi}$ representing a chess position square-by-square. Besides a Boolean-level formula for $WhiteIsMated$, we would also need a predicate $IsLegalMove(\vec{\pi}, \overrightarrow{bm}_1, \vec{\pi}')$ where we need duplicate copy $\vec{\pi}'$ of the variables in $\vec{\pi}$ to represent the position after Black's first move. And so on with an invocation of $IsLegalMove(\vec{\pi}', \overrightarrow{wm}_1, \vec{\pi}'')$ up until the final checkmate position. The relevant analogy from chess to Turing machines is that our main theorem will involve how IDs work like their "positions".

The mate-in-3 formula counts as having $k = 5$ alternations. A mate-in-4 would be 7 alternations, and so on. It seems like if we just wanted to define "Black can give checkmate" we would need infinitely many quantifiers and variables to handle the possibility of arbitrarily long checkmates, but here is where the "restricted space" of a concrete 8×8 chessboard comes in. Owing to various considerations including the "fifty move rule" there is an upper limit on the length of a possible checkmate and hence on the size of the formula. Controlling how the formula size grows with space and time usage is the key to the proof of our main theorem today.

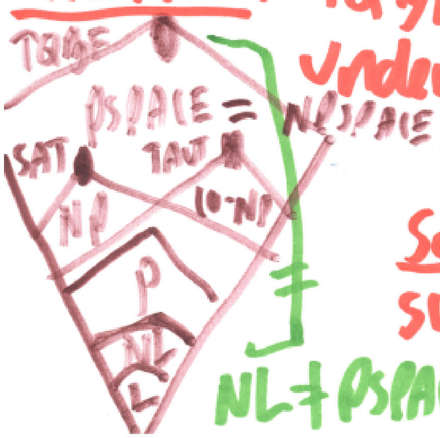
Let **TQBF** denote the language of **true** QBFs (in prenex form).

Note: Misnomers and variant usages abound: When all variables in ψ are quantified---as represented above---ψ should really be called a quantified Boolean **sentence**. Only a sentence can be true or false; strictly speaking, the word *satisfiable* applies whenever there is at least one **free** (i.e., unquantified) variable and there is a way to make the formula true. When all assignments to the free variables $\vec{x}$ make the formula true then ψ is often called "true" although properly it is the QBF $(\forall \vec{x})\psi$ that is true. The language of true QBFs is often (confusingly) called just **QBF**. The non-quantifier body ϕ of a QBF in prenex form is called its **matrix**.

PSPACE-completeness of TQBF

The above already shows $SAT \le_m^{\log} TQBF$ and $TAUT \le_m^{\log} TQBF$. Thus **TQBF** cannot be in NP or in co-NP unless NP $=$ co-NP. We will locate it at a higher completeness level, that of polynomial space, PSPACE.

Theorem: TQBF is complete for PSPACE under $\le_m^{\log}$ reductions



The proof will also show:

Savitch's Theorem For any "reasonable" $s(n) \ge \log_2 n$, NSPACE$[s(n)]$ is in DSPACE$[s(n)^2]$.

Theorem: **TQBF** is complete for PSPACE under $\le_m^{\log}$.

Proof. First, we need to show that **TQBF** belongs to PSPACE. This is one place where limiting QBFs to prenex form comes in handy.

$$\psi = (\exists x_1)(\forall_2 x_2)(\exists_3 x_3) \cdots (\forall_{n-1} 1)(\exists_n 0)\phi(x_1, x_2, \ldots 1, 0),$$

Proof: TQBF $\in$ DSPACE$[O(N)]$ because we can do brute-force evaluation of the qbf using only the space occupied by all the variables

(handwave)

Now let any $A \in$ PSPACE be given. Take a DTM M that accepts A using space $O(n^k)$ for some $k \ge 1$. Given any x, we need to produce a QBF ψ_x that is true $\iff x \in A$.

Actually, our proof will not care whether we take an NTM N instead, and will work for any general space bound $s(n) \ge \log n$---this is how we will deduce Savitch's theorem from the proof.

Key idea: Think again of G_x, the ID graph with edges (I, J) s.t. $I \vdash_N J$. Then $x \in A \Leftrightarrow G_x$ has a path of length $2^{O(s(n))}$ $\leq |G_x|$ from the start ID $I_0(x)$ to {an / the} accepting ID I_f.

$n = |x|$ Put $2^r = \text{"} 2^{O(s(n))} \text{"}$, so $r = r(n)$.

For $0 \leq j \leq r$, define $\Phi_j(I, K)$ to hold iff

$$I \vdash_N^* K \text{ in at most } 2^j \text{ steps}$$

So: $x \in A \Leftrightarrow \Phi_r(I_0(x), I_f)$.

$$\Leftrightarrow (\exists J)\ \Phi_{r-1}(I_0(x), J) \wedge \Phi_{r-1}(J, I_f)$$

Generally,

$$\Phi_j(I,K) \Leftrightarrow (\exists J)\, \Phi_{j-1}(I,J) \wedge \Phi_{j-1}(J,K).$$

$$\Leftrightarrow (\exists J)(\forall I', J'):$$

$$\left[\begin{array}{l} (I' = I \wedge J' = J) \\ \vee\ (I' = J \wedge J' = K) \end{array}\right] \Rightarrow \Phi_{j-1}(I', J').$$

This is a single branch recursion. At bottom

$$\Phi_0(I,K) =_{def} I = K \vee I \overset{1}{\vdash}_N K.$$

Total size is $r \times |\Phi_0(I,K)| = O(r^2)$.